Izrada Android aplikacije za prijenos video signala

Romšić, Danijel

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:658917

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij

IZRADA ANDROID APLIKACIJE ZA PRIJENOS VIDEO SIGNALA

Završni rad

Danijel Romšić

Osijek, 2017.

SADRŽAJ

1. UVOD

Tema ovog završnog rada je izrada Android aplikacije za prijenos video signala koristeći tehniku podatkovnog toka (engl. *streaming*) što znači korištenje kamere pametnog telefona za snimanje video sadržaja te izravno odašiljanje video signala prema poslužitelju koji taj signal treba poslužiti na način da se u web pregledniku, na određenoj adresi, isti sadržaj prikazuje korisniku.

Za izradu aplikacije korišteno je službeno razvojno okruženje Android Studio, unutar kojega se koristi programski jezik Java i jezik za označavanje podataka XML (engl. *Extensible Markup Language).* Kao rješenje problema poslužitelja odabran je poslužiteljski medijski softver Wowza Streaming Engine koji nudi gotova rješenja pri izradi aplikacija za prijenos video signala. U razvoju aplikacija koje pružaju korisniku ovakve mogućnosti susrećemo se sa specifičnim protokolima u komunikaciji između klijenta i poslužitelja.

U nastavku završnog rada slijedi opis po poglavljima: operativni sustav Android, tehnologije korištene u izradi aplikacije unutar razvojnog okruženja Android Studio s prikazom Java i XML kôda, instalacija i postavljanje poslužiteljskog medijskog softvera, specifični protokoli za komunikaciju između klijenta i poslužitelja, te prikaz web stranice na kojoj korisnik pregledava sadržaj.

1.1. Zadatak završnog rada

Opisati mogućnosti i način pristupa ugrađenoj kameri „pametnog“ telefona i/ili tableta. Opisati sve potrebne korake i prikazati ih na primjeru aplikacije koja koristi ugrađenu kameru za distribuciju video signala prema drugim uređajima na lokalnoj ili globalnoj računalnoj mreži. Također opisati moguće načine distribucije razvijene aplikacije.

2. TEHNOLOGIJE KORIŠTENE U IZRADI APLIKACIJE

Prilikom izrade aplikacije korištene su slijedeći alati i tehnologije: Android Studio, Android SDK, programski jezik Java, jezik za označavanje podataka, Wowza GoCoder SDK, Wowza Streaming Engine, nekoliko specifičnih internet protokola koji omogućuju uspješan prijenos i kontrolu podataka.

2.1. Android Studio

Android Studio je službeno integrirano razvojno okruženje za Android platformu. Objavljen je 16. 5. 2013. na Google I/O konferenciji. Baziran je na JetBrains IntelliJ IDEA softveru i dizajniran specifično za Android razvoj.

Dostupan je na Windows, macOS i Linux operativnim sustavima. Neke od mogućnosti koje pruža su: Gradle-bazirana podrška izgradnje, restrukturiranje i brzo ispravljanje koda, Lint alate za zahvaćanje performansi, koristivosti, kompatibilnosti verzija i drugih problema, ProGuard integracija i mogućnosti korisničkih upisivanja u aplikacije, Android virtualni uređaj (engl. *emulator*) za pokretanje i ispravljanje pogrešaka u aplikaciji, te mnoge druge.

Android Studio razvojno okruženje sadrži Android SDK (Software Development Kit) upravitelja (engl. *manager*) [2][3].

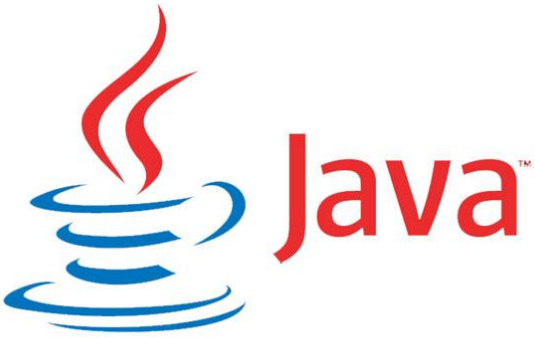
2.2. Android SDK

Android SDK (engl. *Software Development Kit*) uključuje opsežan set razvojnih alata. To uključuje ispravljač pogrešaka (engl. *debugger*), biblioteke, virtualni uređaj, dokumentaciju, primjere kôda i lekcije (engl. *tutorial*). Podržava starije verzije Androida u slučaju da programeri žele prilagoditi svoju aplikaciju starijim uređajima. Razvojni alati su komponente koje je moguće skinuti, pa nakon što je skinuta najnovija verzija platforme, moguće je dodatno skinuti i starije verzije platforme i alata radi testiranja kompatibilnosti [4].

2.3. Programski jezik Java

Java je objektno orijentirani programski jezik razvijen u tvrtci Sun Microsystem. Glavna prednost nad ostalim programskim jezicima njegova je neovisnost o operativnom sustavu na kojem se izvodi ako za njega postoji JVM (engl. *Java Virtual Machine).* To je postignuto tako što je Java programski kôd preveden u reprezentaciju zvanu Java bajtkod (engl. *bytecode*), umjesto izravno na arhitekturno specifičan strojni jezik. Java bajtkod instrukcije su analogne strojnom jeziku. Standardne biblioteke omogućuju generički način pristupa mogućnostima kao što su grafika, korištenje niti i umrežavanje specifičnih za domaćina (engl. *host*). Java koristi automatsko skupljanje otpadaka za upravljanje memorijom u životnom ciklusu objekta, što omogućuje programerima uštedu vremena jer ne moraju brinuti o oslobađanju memorije. Na sintaksu programskog jezika Java uvelike je utjecao programski jezik C++, no za razliku od njega, Java ne podržava opterećivanje operatora, niti višestruko nasljeđivanje klasa, osim za sučelje.

Java je jedan od najsigurnijih, najraširenijih i najkorištenijih programskih jezika, osobito za klijent-poslužitelj web aplikacije. Java je osnovni programski jezik za programiranje na Android platformi. Iako Android, izgrađen na Linux jezgri, uvelike pisan u programskom jeziku C, koristi alternativni softver Android SDK za izradu Android aplikacija. Bajtkod jezik podržan od Android SDK nekompatibilan je s Java bajtkodom i pokreće se na vlastitom virtualnom stroju, optimiziranom za uređaje s nižom memorijom. Android ne omogućava potpunu Java SE standardnu biblioteku, iako Android SDK uključuje neovisnu implementaciju njegovog većeg dijela [5].



Sl. 2.1. Java logotip (preuzeto s www. airbrake.io)

2.4. XML

XML (engl. *Extensible Markup Language*) je jezik za označavanje podataka, odnosno opisni jezik koji je zamišljen tako da bude lako čitljiv i ljudima i računalima. Podaci su označeni tako što su uokvireni unutar oznaka (engl. *tag*) te je na taj način definirana njihova prezentacija. Konkretno, na Android platformi opisan je izgled korisničkog sučelja (boja, veličina i pozicija pojedinih elemenata) te ponašanje elemenata ukoliko korisnik vrši interakciju s njima [6].

2.5. Wowza GoCoder SDK

Wowza GoCoder SDK proširivi je i višeplatformski API (engl. *Application programming interface)* za snimanje i podatkovni tok audio i video sadržaja pojednostavljivanjem razvoja mobilnih aplikacija i korisnički kreiranim rješenjima jednostavnim odašiljanjem sadržaja na Wowza Streaming Engine.

Uključuje podršku za širok raspon Android i iOS uređaja, razne postavke kodiranja, kao i snimanje u 4K rezoluciji. Omogućuje korištenje prednje i stražnje kamere mobilnog uređaja, korištenje svjetla, arhiviranje sadržaja lokalno na uređaju, pregled i kontrolu strujanja. Spajanje na mrežu je moguće preko 3G i 4G mobilne mreže te preko Wi-Fi mreže.

2.6. Wowza Streaming Engine

Wowza Streaming Engine (prethodno poznat kao Wowza Media Server) ujedinjeni je poslužiteljski medijski softver za podatkovni tok razvijen od strane tvrtke Wowza Media Systems. Poslužitelj se koristi za podatkovni tok video sadržaja na zahtjev, audio sadržaja te bogatih internet aplikacija preko IP mreža za stolna, prijenosna i tablet računala, mobilne uređaje, TV setove koji su spojeni na internet, igraće konzole i ostale uređaje koji su mrežno spojivi. Poslužitelj je Java aplikacija koju je moguće razviti na većini operativnih sustava.

Wowza Streaming Engine može imati simultani podatkovni tok različitim tipovima reprodukcijskih klijenata i uređaja, uključujući Adobe Flash Player. Wowza Streaming Engine kompatibilan je sa standardnim protokolima za tok podataka. To uključuje RTMP, HDS, HLS, MPEG DASH, RTSP, Smooth Streaming itd. Također podržava dolazeći podatkovni tok preko RTSP protokola s Android i iOS uređaja koji pokreću Wowza GoCoder mobilnu aplikaciju [7].

Sl. 2.2. Grafički prikaz mogućnosti korištenja Wowza Streaming Engine softvera

2.7. Specifični protokoli

Da bi se ovakav tip aplikacije uspješno realizirao i da bi komunikacija između uređaja koji koristi operativni sustav Android i krajnje odredišne točke što je internetska stranica potrebno je uvesti te iskoristiti nekoliko specifičnih protokola. Najznačajniji od tih protokola koji se koriste, a opisani su u nastavku teksta su RTSP protokol i RTMP protokol.

2.7.1. RTSP (engl. Real -Time Streaming Protocol) protokol

RTSP je protokol na aplikacijskoj razini korišten za prijenos stvarnovremenskih medijskih podataka. Protokol se koristi za uspostavu i kontrolu sesije između krajnjih točaka služeći kao mrežni upravljač za vremenski sinkroniziran podatkovni tok kontinuiranih medija kao što su audio i video sadržaj. RTSP je razvijen od strane RealNetworks, Netscape i Columbia University. Standardiziran je od Multiparty Multimedia Session Control Working Group-a (MMUSIC WG),

Internet Engineering Task Force-a (IETF) i objavljen 1998. godine kao RFC 2326, a 2016. godine kao njegova zamjena objavljen je RTSP 2.0.

Iako je po nekim značajkama sličan HTTP (engl. *HyperText Transfer Protocol*) protokolu, RTSP definira kontrolne sekvence korisne za kontrolu multimedijskog reproduciranja. Dok je HTTP bez stanja, RTSP ima stanje, a identifikator je korišten kada je potrebno praćenje konkurentnih sesija. Kao i HTTP, RTSP koristi TCP (engl. *Transmission Control Protocol*) za održavanje veze između krajnjih točaka te, dok je većina kontrolnih poruka poslano od strane klijenta prema poslužitelju, neke naredbe putuju i u suprotnom smjeru. Također koriste jednaku URL strukturu za opisivanje objekata, no RTSP sadrži i dodatna zaglavlja kao što su opisno zaglavlje, zaglavlje s postavkama te zaglavlje za reprodukciju.

Osnovni koraci uključeni u proces dostavljanja audio/video podatkovnih tokova su sljedeći:

1. Klijent uspostavlja TCP konekciju poslužiteljima, uobičajeno na ulazu 554, poznatom kao ulaz za RTSP

2. Klijent tada započinje slanje serije naredbi RTSP zaglavlja koja su obznanjena od strane poslužitelja. Unutar RTSP naredbi, klijent će opisati poslužitelju detalje o zahtjevima sesije kao što su verzije podržanog RTSP-a, korišteni prijenos za protok podataka i bilo koju relevantnu informaciju UDP (engl. *User Datagram Protocol*) ili TCP ulaza. Ta informacija predana je koristeći opisno zaglavlje i zaglavlje postavki i argumentirana je na poslužiteljev odaziv s identifikacijom sesije koju klijent i bilo koji posrednički uređaji mogu koristiti za identifikaciju strujanja u nadolazećim izmjenama.

3. Kada je pregovaranje transportnih parametara završeno, klijent će *PLAY* naredbom narediti poslužitelju izvršavanje dostave RTP toka podataka.

4. Kada klijent odluči završiti protok, naredba *TEARDOWN* je poslana zajedno s identifikacijom sesije naređujući poslužitelju da prekine RTP dostavu asociranu pomoću te identifikacije [8].

2.7.2. RTMP (engl. Real-time Messaging Protocol) protokol

RTMP protokol prvobitno služi za prijenos audio i video sadržaja pri visokim brzinama te podataka između Flash Playera i poslužitelja. Razvila ga je tvrtka Macromedia koja je sada u vlasništvu tvrtke Adobe. Specifikacije protokola samo su djelomično izašle za javno korištenje, a prema njima RTMP ima više varijacija (RTMPS, RTMPT, RTMPE, RTMPTE). Uobičajeni RTMP koristi TCP port pod brojem 1935.

Svrha RTMP protokola jest izbjegavanje latentnosti u komunikaciji te dostavljanje audio i video podatkovnog toka, odvajajući ih u fragmente i čineći ih naizmjeničnim i multipleksiranim preko jedne konekcije. Multipleksiranje se izvršava na razini paketa s RTMP paketima preko nekoliko različitih kanala, gdje se izmjenjuju tako da se svakom kanalu osigura njegova širina, latencija te da zadovoljava potrebe kvalitete usluge. RTMP protokol definira nekoliko virtualnih kanala na koje paketi mogu biti poslani i primani, a koji se izvršavaju neovisno. Tijekom regularne RTMP sesije, nekoliko kanala može biti simultano aktivno u svakom trenutku.

Primjeri podataka koji se mogu prenositi su: prije snimljeni podaci, video podaci uživo, audio podaci uživo, tekstualne poruke, vektori i slično.

Protokol funkcionira tako da Flash Player prvo kontaktira Flash medijskog poslužitelja, nakon čega se uspostavlja konekcija. Flash Player zatraži specifičan podatkovni tok te jednom, kada medijski poslužitelj zaprimi zahtjev, podatkovni tok se šalje izravno Flash Player-u preko RTMP konekcije. Jednak podatkovni tok može biti poslan bilo kojem broju klijenata koji su ga zatražili, računajući da je širina prijenosnog kanala dovoljno dobra na obje strane za neometano dostavljanje svih simultanih podatkovnih tokova [9].

2.8. Operativni sustav Android

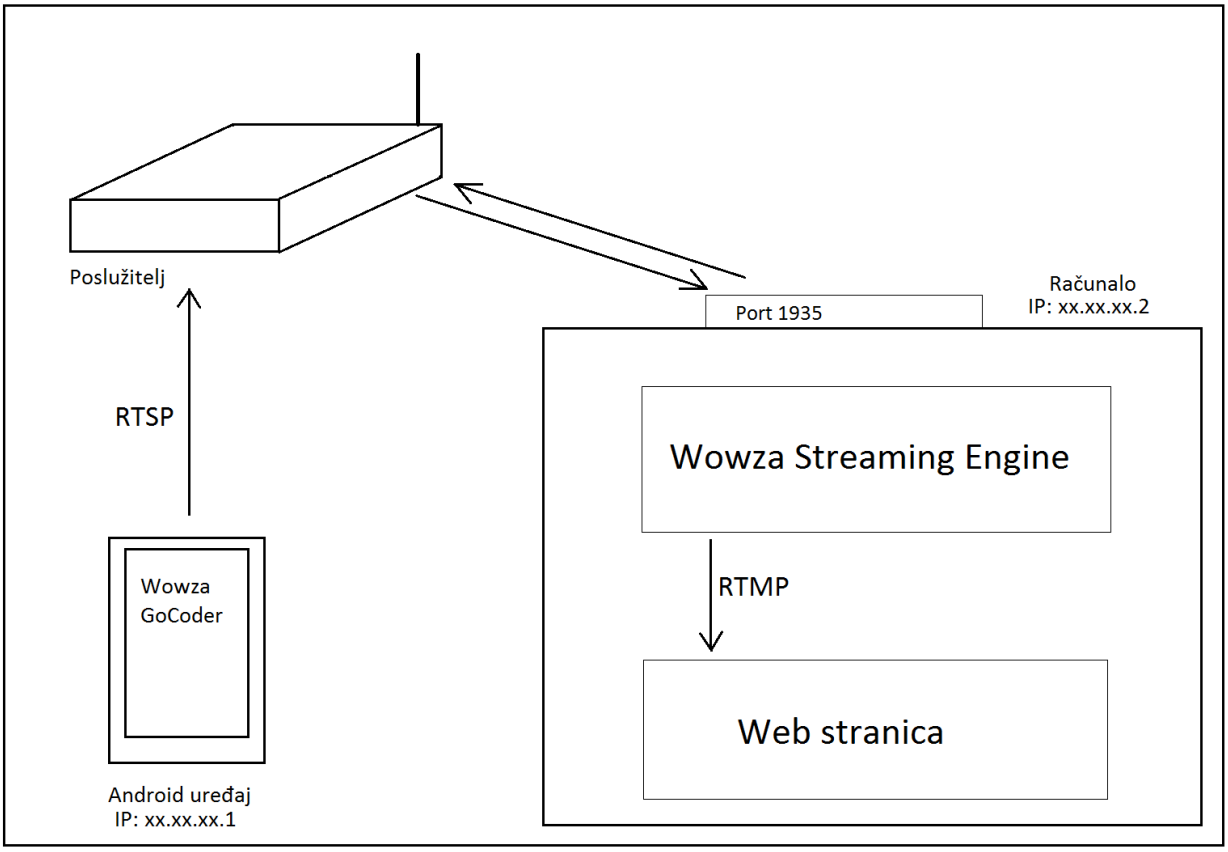
Android je mobilni operativni sustav razvijen u Google-u, baziran na Linux-ovoj jezgri i dizajniran primarno za pametne telefone i tablet računala. Androidovo korisničko sučelje uglavnom je bazirano na direktnoj manipulaciji, koristeći dodirne geste koje odgovaraju akcijama iz stvarnog svijeta, zajedno s virtualnom tipkovnicom za tekstualni unos. Unutarnji hardver, kao što su akcelerometar, žiroskop i senzori približnosti korišteni su u nekim aplikacijama za reakciju na dodatne korisnikove akcije.

Androidov izvorni kod je izdan od Google-a pod licencom otvorenog kôda. Android je popularan u tehnološkim kompanijama koje zahtijevaju jeftin, gotov i prilagođavajući operativni sustav za visokotehnološke uređaje [1].

Sl. 2.3. Android logotip (preuzeto s www.android.com)

3. REALIZACIJA PROJEKTA

Projekt je realiziran unutar razvojnog okruženja Android Studio korištenjem programskog jezika java.



Slika 3.1. Shematski prikaz projekta

3.1. Android Studio

Unutar razvojnog okruženja Android Studio koristimo Java kôd i XML kôd.

3.1.1. Java kôd

Unutar datoteke glavna aktivnost (engl. *main activity)* Android Studio projekta upisana je programska logika i niz metoda koje omogućuju realiziranje projekta. Neke od ovih metoda su navedene u nastavku. Autori kompletnog kôda aplikacije su programeri tvrtke Wowza Media

Systems (pod opaskom da se nigdje ne spominje točno ime i prezime). Pošto se radi o demonstracijskom primjeru, postoje ograničenja a to se prvenstveno odnosi na to da se pri preuzimanju demonstracijskog kôda dobije licenca koja se upisuje unutar Java koda na za to predviđeno mjesto te je demonstracijsku aplikaciju moguće uspješno koristiti samo pri jednoj instalaciji, što znači da ako istu instaliramo po drugi put na neki drugi uređaj ona neće biti funkcionalna.

Da bismo omogućili prikaz preko cijelog zaslona dodajemo `onWindowFocusChanged()` metodu unutar `MainActivity` klase:

```
    @Override
public void onWindowFocusChanged(boolean hasFocus) {
    super.onWindowFocusChanged(hasFocus);
    View rootView =
getWindow().getDecorView().findViewById(android.R.id.content);
    if (rootView != null)
        rootView.setSystemUiVisibility(
                View.SYSTEM_UI_FLAG_LAYOUT_STABLE
                        |
View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
                        | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
                        | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
                        | View.SYSTEM_UI_FLAG_FULLSCREEN
                        | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY);
}
```

Definiranje svojstava aplikacije dodavanjem članskih varijabli klasi *MainActivity* unutar datoteke *MainActivity.java*:

```
    public class MainActivity extends AppCompatActivity

        implements WZStatusCallback, View.OnClickListener {
    // The top level GoCoder API interface
    private WowzaGoCoder goCoder;
    // The GoCoder SDK camera view
    private WZCameraView goCoderCameraView;
    // The GoCoder SDK audio device
    private WZAudioDevice goCoderAudioDevice;
```

```
    // The GoCoder SDK broadcaster
    private WZBroadcast goCoderBroadcaster;
    // The broadcast configuration settings
    private WZBroadcastConfig goCoderBroadcastConfig;
    // Properties needed for Android 6+ permissions handling
    private static final int PERMISSIONS_REQUEST_CODE = 0x1;
    private boolean mPermissionsGranted = true;
    private String[] mRequiredPermissions = new String[] {
            Manifest.permission.CAMERA,
            Manifest.permission.RECORD_AUDIO
    };
```

Kako se za potrebe izrade ovog završnog rada koristi besplatna verzija GoCoder SDK, potrebno je registrirati licencu i inicijalizirati SDK dodajući slijedeće linije kôda unutar *onCreate()* metode u klasi *MainActivity*:

```
     // Initialize the GoCoder SDK

    goCoder = WowzaGoCoder.init(getApplicationContext(), "GOSK-
C243-0103-30B8-F388-51B8");

if (goCoder == null) {
    // If initialization failed, retrieve the last error and
display it
    WZError goCoderInitError = WowzaGoCoder.getLastError();
    Toast.makeText(this,
            "GoCoder SDK error: " +
goCoderInitError.getErrorDescription(),
            Toast.LENGTH_LONG).show();
    return;
}
```

Slijedi provjera aplikacijskih dozvola dodavanjem *onResume()* metode i dodatne *onRequestPermissionsResults()* metode klasi *MainActivity* radi provjere da li je korisnik dozvolio aplikaciji pravo pristupa kameri i mikrofonu uređaja:

```
@Override

protected void onResume() {

    super.onResume();
    // If running on Android 6 (Marshmallow) or above, check to
see if the necessary permissions
    // have been granted
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        mPermissionsGranted = hasPermissions(this,
mRequiredPermissions);
        if (!mPermissionsGranted)
            ActivityCompat.requestPermissions(this,
mRequiredPermissions, PERMISSIONS_REQUEST_CODE);
    } else
        mPermissionsGranted = true;
    // Start the camera preview display
    if (mPermissionsGranted && goCoderCameraView != null) {
        if (goCoderCameraView.isPreviewPaused())
            goCoderCameraView.onResume();
        else
            goCoderCameraView.startPreview();
    }
}
// Callback invoked in response to a call to
ActivityCompat.requestPermissions() to interpret
// the results of the permissions request

@Override
public void onRequestPermissionsResult(int requestCode, String
permissions[], int[] grantResults) {
    mPermissionsGranted = true;
    switch (requestCode) {
        case PERMISSIONS_REQUEST_CODE: {
            // Check the result of each permission granted
            for(int grantResult : grantResults) {
                if (grantResult !=
PackageManager.PERMISSION_GRANTED) {
                    mPermissionsGranted = false;
                }
            }}}}
//
// Utility method to check the status of a permissions request
for an array of permission identifiers
//
private static boolean hasPermissions(Context context, String[]
permissions) {
```

```
    for(String permission : permissions)
        if (context.checkCallingOrSelfPermission(permission) !=
PackageManager.PERMISSION_GRANTED)
            return false;
    return true;}
```

Slijedeći korak je konfiguriranje i pokretanje pregleda kamere tako što se asocira pogled (engl. *view*) definiran unutar rasporeda aktivnosti s odgovarajućim članom klase *WZCameraView* i kreiranjem instance ulaznog audio uređaja dodavajući *onCreate()* metodi slijedeće:

```
    // Associate the WZCameraView defined in the U/I layout with
the corresponding class member

goCoderCameraView = (WZCameraView)
findViewById(com.wowza.gocoder.sdk.sampleapp.R.id.camera_preview
);
// Create an audio device instance for capturing and
broadcasting audio
goCoderAudioDevice = new WZAudioDevice();
```

Te za uključivanje pregleda kamere dodavanjem *onResume()* metodi slijedećeg:

```
    // Start the camera preview display

if (mPermissionsGranted && goCoderCameraView != null) {
    if (goCoderCameraView.isPreviewPaused())
        goCoderCameraView.onResume();
    else
        goCoderCameraView.startPreview();
}
```

Zatim slijedi konfiguracija odašiljanja podatkovnog toga dodavajući adresu domaćina (*hostAddress*), broj ulaza (*portNumber*), naziv aplikacije (*applicationName*) i naziv podatkovnog toka (*streamName)* za Wowza Streamg Enginein poslužitelja *onCreate()* metodi *MainActivity* klase:

```
// Create a broadcaster instance

goCoderBroadcaster = new WZBroadcast();

// Create a configuration instance for the broadcaster
```

```
goCoderBroadcastConfig = new
WZBroadcastConfig(WZMediaConfig.FRAME_SIZE_1920x1080);
// Set the connection properties for the target Wowza Streaming
Engine server or Wowza Cloud account
goCoderBroadcastConfig.setHostAddress("192.168.5.13");
goCoderBroadcastConfig.setPortNumber(1935);
goCoderBroadcastConfig.setApplicationName("StreamApp");
goCoderBroadcastConfig.setStreamName("myStream");
// Designate the camera preview as the video broadcaster
goCoderBroadcastConfig.setVideoBroadcaster(goCoderCameraView);
// Designate the audio device as the audio broadcaster
goCoderBroadcastConfig.setAudioBroadcaster(goCoderAudioDevice);
```

Slijedi dodavanje povratnih poziva nadzoru odašiljanja dodajući metode definirane u *WZStatusCallback* sučelju klasi *MainActivity*:

```
// The callback invoked upon changes to the state of the
steaming broadcast
//
@Override
public void onWZStatus(final WZStatus goCoderStatus) {
    // A successful status transition has been reported by the
GoCoder SDK
    final StringBuffer statusMessage = new
StringBuffer("Broadcast status: ");
    switch (goCoderStatus.getState()) {
        case WZState.STARTING:
            statusMessage.append("Broadcast initialization");
            break;
        case WZState.READY:
            statusMessage.append("Ready to begin streaming");
            break;
        case WZState.RUNNING:
            statusMessage.append("Streaming is active");
            break;
        case WZState.STOPPING:
            statusMessage.append("Broadcast shutting down");
            break;
        case WZState.IDLE:
            statusMessage.append("The broadcast is stopped");
            break;
        default:
            return;
    }
```

```
    // Display the status message using the U/I thread
    new Handler(Looper.getMainLooper()).post(new Runnable() {
        @Override
        public void run() {
            Toast.makeText(MainActivity.this, statusMessage,
Toast.LENGTH_LONG).show();
        }
    });
}
//
// The callback invoked when an error occurs during a broadcast
//
@Override
public void onWZError(final WZStatus goCoderStatus) {
    // If an error is reported by the GoCoder SDK, display a
message
    // containing the error details using the U/I thread
    new Handler(Looper.getMainLooper()).post(new Runnable() {
        @Override
        public void run() {
            Toast.makeText(MainActivity.this,
                    "Streaming error: " +
goCoderStatus.getLastError().getErrorDescription(),
                    Toast.LENGTH_LONG).show();
        }
    });}
```

Slijedi kôd za konfiguraciju gumba odašiljanja koji pokreće i zaustavlja podatkovni tok kada je pritisnut. Dodavanje *View.onClickListener* sučelja definiciji klase *MainActivity*:

```
    public class MainActivity extends AppCompatActivity

        implements WZStatusCallback, View.OnClickListener {
```

Zatim dodavanje *onClick*() rukovoditelja događaja:

```
// The callback invoked when the broadcast button is pressed
//
@Override
public void onClick(View view) {
  // return if the user hasn't granted the app the necessary
permissions
  if (!mPermissionsGranted) return;
  // Ensure the minimum set of configuration settings have been
specified necessary to
```

```
  // initiate a broadcast streaming session
  WZStreamingError configValidationError =
goCoderBroadcastConfig.validateForBroadcast();
  if (configValidationError != null) {
    Toast.makeText(this,
configValidationError.getErrorDescription(),
Toast.LENGTH_LONG).show();
  } else if (goCoderBroadcaster.getStatus().isRunning()) {
    // Stop the broadcast that is currently running
    goCoderBroadcaster.endBroadcast(this);
  } else {
    // Start streaming
    goCoderBroadcaster.startBroadcast(goCoderBroadcastConfig,
this);
  }
}
```

Te asociranje *onClick()* rukovoditelja događaja s gumbom za odašiljanje dodajući slijedeće *onCreate()* metodi:

```
    // Associate the onClick() method as the callback for the
broadcast button's click event

Button broadcastButton = (Button)
findViewById(com.wowza.gocoder.sdk.sampleapp.R.id.broadcast_butt
on);
broadcastButton.setOnClickListener(this);
```

3.1.2. XML kôd

Unutar *AndroidManifest.xml* datoteke potrebno je unijeti dozvole za korištenje nekih od mogućnosti uređaja koje će aplikacija koristiti:

```
<uses-permission android:name="android.permission.CAMERA" />

<uses-permission android:name="android.permission.RECORD_AUDIO"
/>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.FLASHLIGHT" />
<uses-permission
android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
```

Također dodajemo *android:configChanges* svojstvo definiciji *activity* elementa u istoj datoteci:

```
<activity android:name=".MainActivity"

android:configChanges="orientation|keyboardHidden|screenSize">
```

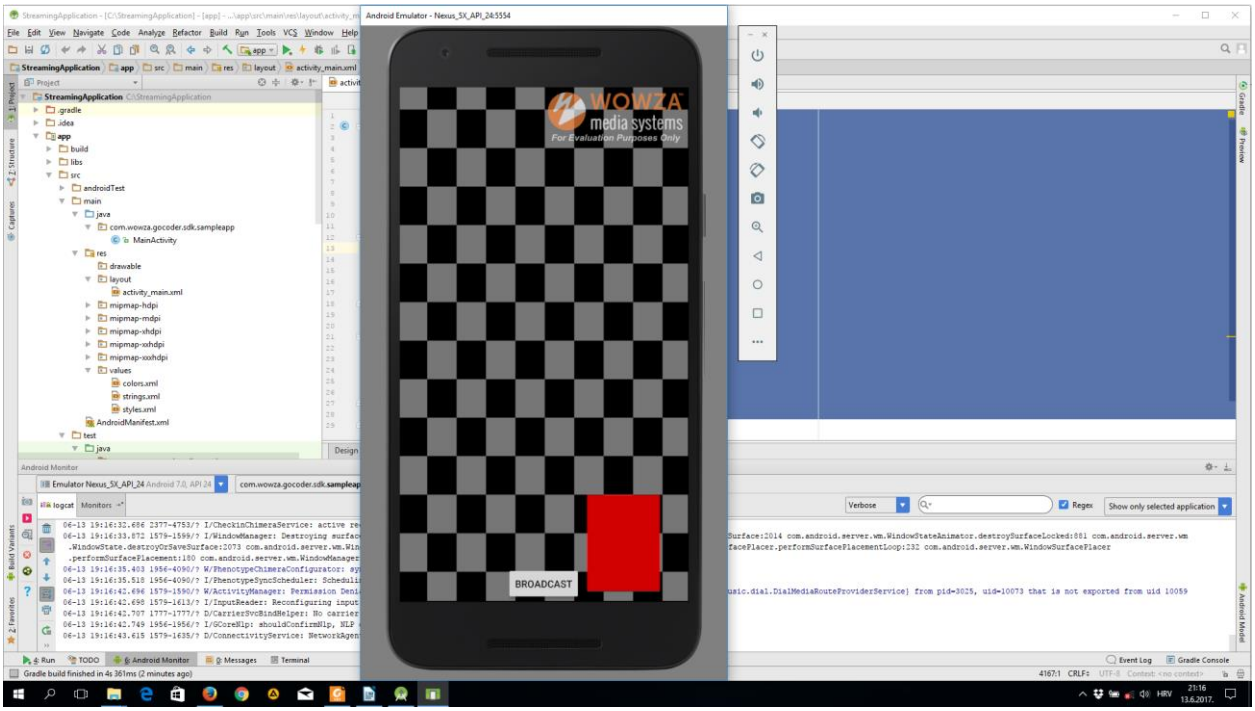
Izgled korisničkog sučelja je definiran unutar datoteke *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:wowza="http://schemas.android.com/apk/res-auto"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <!-- The camera preview display -->
    <com.wowza.gocoder.sdk.api.devices.WZCameraView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/camera_preview"
        wowza:scaleMode="fill"
        wowza:defaultCamera="back"
        wowza:frameSizePreset="frameSize1280x720"/>
    <!-- The broadcast button -->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Broadcast"
        android:id="@+id/broadcast_button"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" />
```

3.1.3. Izgradnja i korištenje aplikacije

Nakon što je uređen kôd unutar razvojnog okruženja Android Studio, aplikaciju je potrebno izgraditi (engl. *build)* i instalirati na uređaj koji koristi operacijski sustav Android.



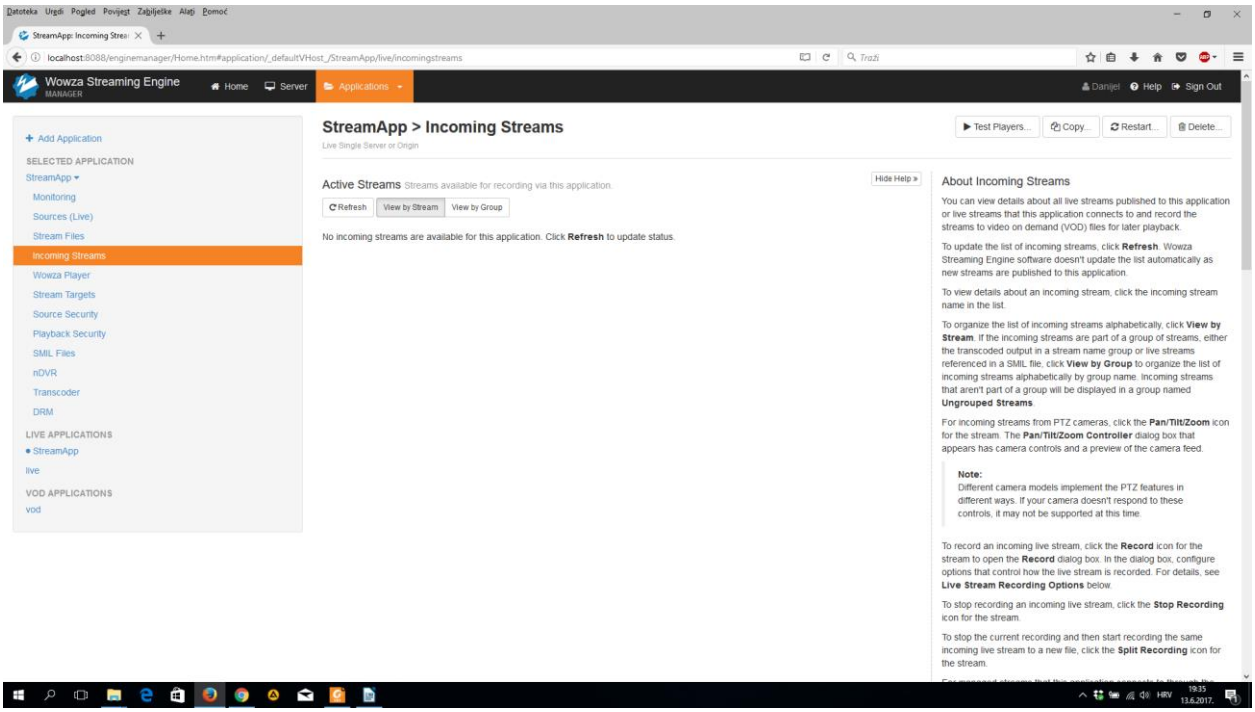
Sl.3.1. Izgled korisničkog sučelja aplikacije prikazan korištenjem virtualnog uređaja

Nakon instalacije aplikacija se pokreće i pritiskom na gumb inicijaliziran je pokušaj spajanja na medijskog poslužitelja. Ukoliko je spajanje uspješno, aplikacija snima video i prosljeđuje medijskom poslužitelju u H.264 formatu.

3.2. Konfiguracija medijskog poslužitelja Wowza Streaming Engine

Prvi korak pri konfiguraciji medijskog poslužitelja je njegova instalacija na računalo. Pri instalaciji je potrebno napraviti registraciju korisnika tj. administratora. Moguće je postaviti ovjeravanje autentičnosti na način da korisnik unese jedinstvenu lozinku te se na taj način onemogući pristup neželjenim dolaznim signalima.

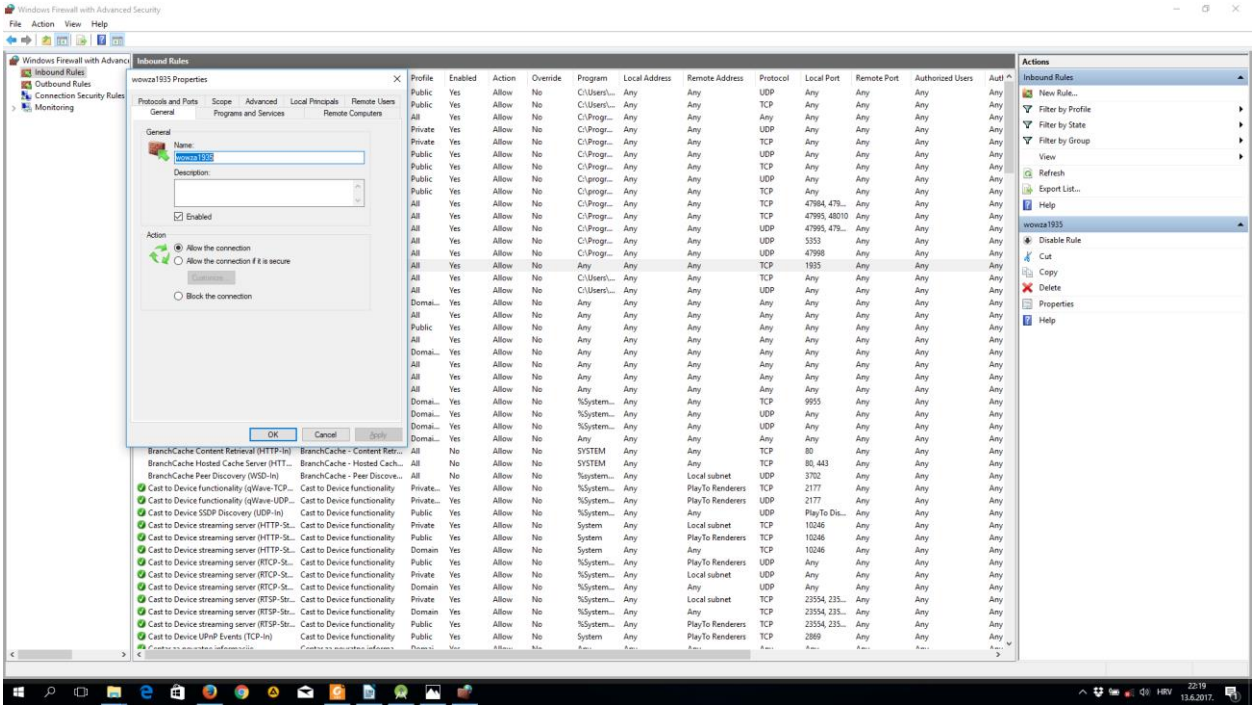
Potrebno je kreirati aplikaciju koja će predstavljati poveznicu između medijskog poslužitelja i Android aplikacije, te ju imenovati kako bi video signal mogao odaslati i prikazati na *Flash player*-u ugrađenom unutar web stranice. Kada Android aplikacija pokrene odašiljanje, dolazni signal bi trebao biti prepoznat od strane medijskog poslužitelja te pritiskom na tipku za osvježavanje statusa bi trebao biti prikazan kao aktivan.



Sl.3.2. Upravitelj medijskog poslužitelja

3.3. Otvaranje TCP porta 1935

Da bi komunikacija između Android uređaja i računala bila moguća potrebno je otvoriti TCP port 1935 na način da se u vatrozidu napravi pravilo koje otvara navedeni port.



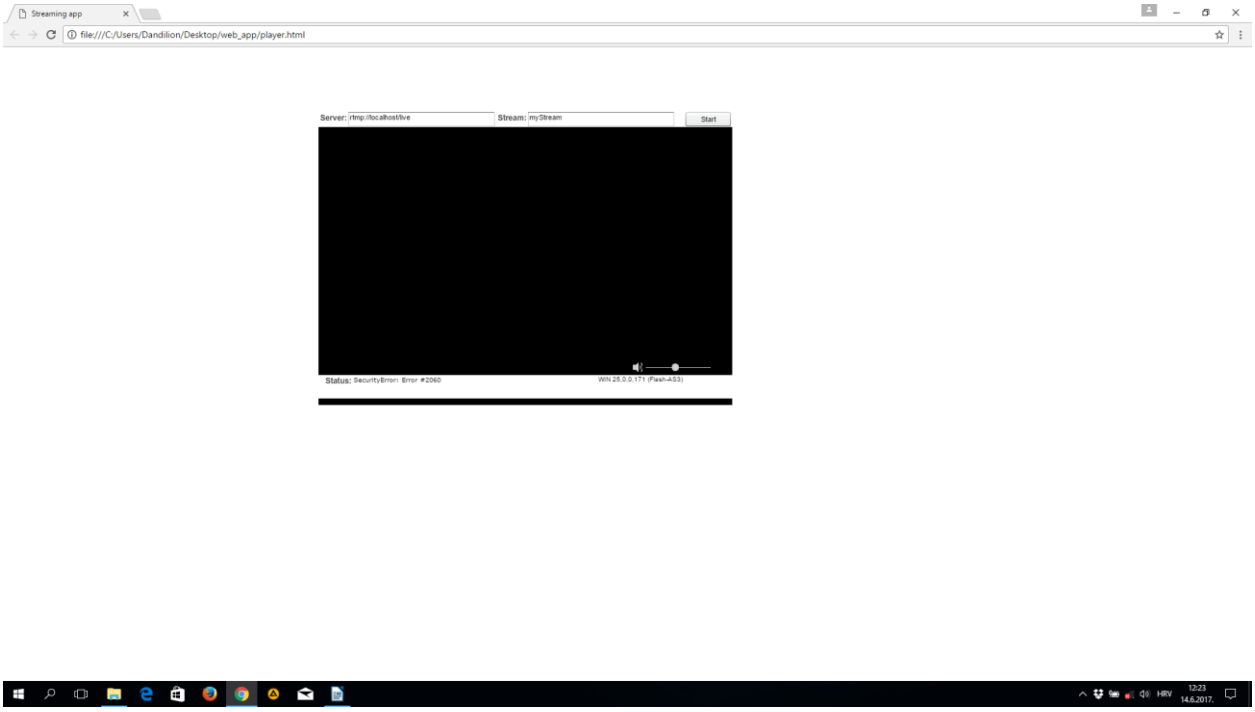
Sl.3.3. Postavke pravila vatrozida

3.4. Prikaz video sadržaja na web stranici

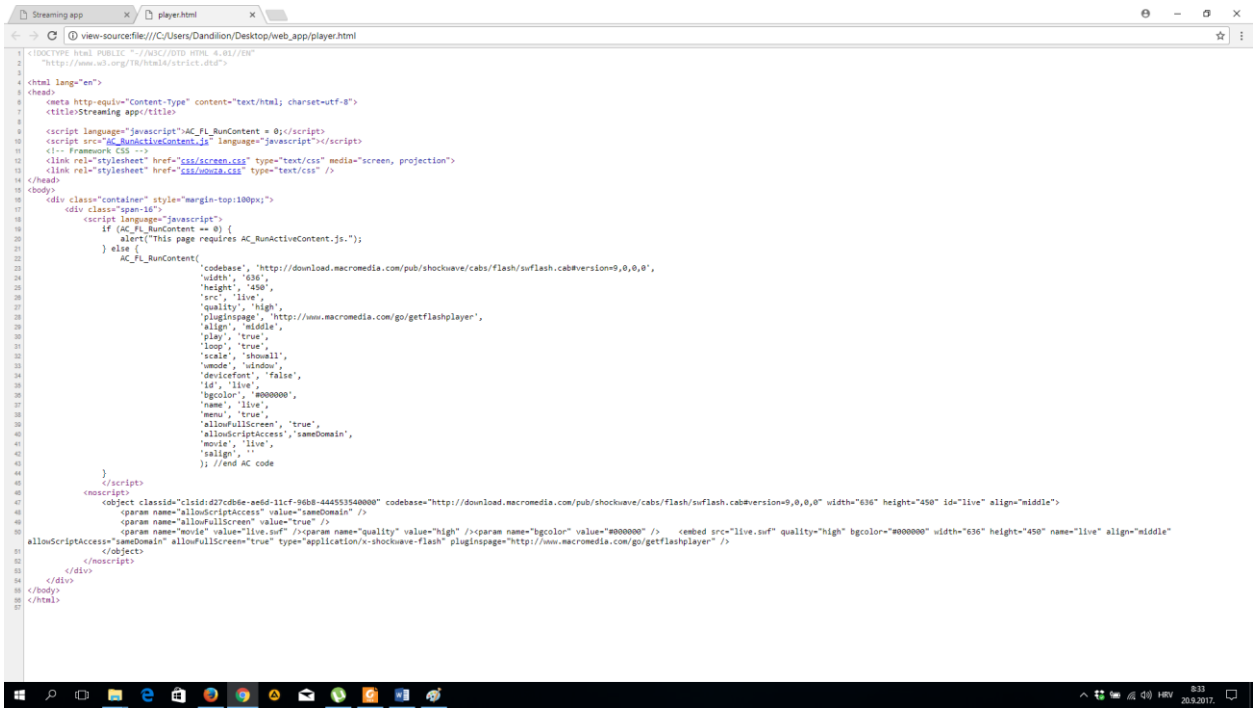
Ako je svaki od prethodno opisanih postupaka izveden ispravno, sve što je potrebno je otvoriti web stranicu s ugrađenim *Flash player*-om, unijeti naziv aplikacije i naziv podatkovnog toka u pripadajuća polja i pritiskom na gumb "Start" reprodukcija video signala je pokrenuta. Korištena web stranica je prethodno izrađena web stranica za potrebe korištenja usluga sličnih aplikacija te je pronađena na razvojnoj platformi GitHub nepoznatog autora. Da bi korištenje web stranice bilo uspješno potreban je prethodno opisan medijski poslužitelj Wowza Streaming Engine.

Za korištenje stranice nije potreban niti jedan web poslužitelj nego samo medijski poslužitelj Wowza Streaming Engine.

U web pregledniku je potrebno otvoriti web stranicu s lokalnog diska.



Sl.3.4. Izgled web stranice za prikaz video signala



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">

<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Streaming app</title>

    <script language="javascript">AC_FL_RunContent = 0;</script>
    <script src="AC_RunActiveContent.js" language="javascript"></script>
    <!-- Framework CSS -->
    <link rel="stylesheet" href="css/screen.css" type="text/css" media="screen, projection">
    <link rel="stylesheet" href="css/HOWIZA.CSS" type="text/css" />
</head>
<body>
    <div class="container" style="margin-top:100px;">
        <div class="span-16">
            <script language="javascript">
                if (AC_FL_RunContent == 0) {
                    alert("This page requires AC_RunActiveContent.js.");
                } else {
                    AC_FL_RunContent(
                                'codebase', 'http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,0,0',
                                'width', '636',
                                'height', '450',
                                'src', 'live',
                                'quality', 'high',
                                'pluginspage', 'http://www.macromedia.com/go/getflashplayer',
                                'align', 'middle',
                                'play', 'true',
                                'loop', 'true',
                                'scale', 'showall',
                                'wmode', 'window',
                                'devicefont', 'false',
                                'id', 'live',
                                'bgcolor', '#000000',
                                'name', 'live',
                                'menu', 'true',
                                'allowFullScreen', 'true',
                                'allowScriptAccess','sameDomain',
                                'movie', 'live',
                                'salign', ''
                                ); //end AC code
                }
                </script>
            <noscript>
                <object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,0,0" width="636" height="450" id="live" align="middle">
                    <param name="allowScriptAccess" value="sameDomain" />
                    <param name="allowFullScreen" value="true" />
                    <param name="movie" value="live.swf" /><param name="quality" value="high" /><param name="bgcolor" value="#000000" />    <embed src="live.swf" quality="high" bgcolor="#000000" width="636" height="450" name="live" align="middle"
allowScriptAccess="sameDomain" allowFullScreen="true" type="application/x-shockwave-flash" pluginspage="http://www.macromedia.com/go/getflashplayer" />
                </object>
            </noscript>
        </div>
    </div>
</body>
</html>
```

Sl.3.5. Izvorni kod web stranice za prikaz video signala

4. ZAKLJUČAK

Operativni sustav Android pruža širok spektar mogućnosti korištenja digitalne tehnologije u područjima komunikacije i informatizacije, izradu aplikacija namjenskog, informativnog i zabavnog sadržaja za potrebe korisnika. Problem izrade i korištenja Android aplikacija je nepotpuna kompatibilnost njihovih ciljanih verzija sustava i mogućnosti uređaja s određenim API nivoom s uređajima nižeg ili višeg API nivoa pa su programeri često u nemogućnosti prilagoditi aplikaciju za svaki od njih.

U području aplikacija za prijenos video signala, osvrnemo li se na ovaj završni rad, ostavljeno je mnogo prostora u daljem razvijanju jer se koristi samo osnovna funkcija i besplatna korisnička licenca. U slučaju da bi bio korišten potpuni kapacitet aplikacije u smislu plaćene licence moguće je postaviti i više podatkovnih tokova s jednog uređaja preko medijskog poslužitelja na više odredišnih točaka kao i uporaba više kamera s istog uređaja ali na način da korisnik uređaja ima mogućnost prebacivanja s glavne tj. stražnje kamere na prednju i obratno. Također je moguće postići da se preko medijskog poslužitelja postavi da se na jednoj web stranici prikazuje više podatkovnih tokova od strane različitih uređaja ali je za to potrebno urediti stranicu da ima više prozora koji reproduciraju video signal.

LITERATURA

[1] Wikipedia: Android, Dostupno na: https://hr.wikipedia.org/wiki/Android_(operacijski_sustav) (rujan 2017)

[2]Službena stranica za razvoj Android aplikacija https://developer.android.com/index.html (rujan 2017)

[3] Wikipedia: Android studio, Dostupno na: https://en.wikipedia.org/wiki/Android_Studio (rujan 2017)

[4] Wikipedia: Android software development, Dostupno na: https://en.wikipedia.org/wiki/Android_software_development (rujan 2017)

[5] Wikipedia: Java (Programski jezik), Dostupno na: https://hr.wikipedia.org/wiki/Java_(programski_jezik) (rujan 2017)

[6] Wikipedia: XML, Dostupno na: https://hr.wikipedia.org/wiki/XML (rujan 2017)

[7] Wowza live streaming software, Dostupno na: https://www.wowza.com/ (rujan 2017)

[8] Wikipedia: RTSP, Dostupno na: https://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol (rujan 2017)

[9] Wikipedia: RTMP, Dostupno na: https://en.wikipedia.org/wiki/Real-Time_Messaging_Protocol (rujan 2017)

SAŽETAK

Ovaj rad predstavlja proces izrade Android aplikacije za prijenos video signala uz korištenje postojeće biblioteke namijenjene za pojednostavljenu izradu i dizajniranje aplikacija kojima je glavna zadaća obavljati pouzdan podatkovni tok.Glavni problem izrade ovakvog tipa aplikacije je prijenos informacija, u ovom slučaju video signala, preko mreže što zahtijeva unajmljivanje i ulaganje financijskih sredstava u mrežne poslužitelje i njihovo održavanje.U ovom radu je demonstriran pojednostavljen i besplatan način korištenja komunikacije između klijenta i poslužitelja na način da je konekcija ostvarena lokalno, odnosno da prijenosno računalo služi kao poslužitelj.

Ključne riječi: Android aplikacija, podatkovni tok

ABSTRACT

ANDROID APPLICATION FOR TRANSFERRING VIDEO SIGNAL

This work represents the process of making Android application for transferring video signal by using existing library used for simplified building and designing applications whose main activity is to perform reliable data streaming.The main problem of building this type of application is transferring data, in this case video signal, across the network which requires renting and paying for network servers and their maintenance.This work demonstrates simplified and free way of using communication between the client and the server in a way that connection is realised locally, which means that laptop is used as a server.

Keywords: Android application, streaming

ŽIVOTOPIS

Danijel Romšić rođen je 6.11.1988. godine u Našicama, Republika Hrvatska. Osnovnu školu pohađao je u Osnovnoj školi Augusta Harambašića u Rakitovici i Donjem Miholjcu. Srednju školu je završio u Donjem Miholjcu za zvanje prodavač. 2011. godine upisao se na stručni studij Elektrotehničkog fakulteta u Osijeku, smjer Informatika.